Fortran Development Tools: Providing a Roadmap for Application Development on Advanced Computer Architectures

Craig E. Rasmussen, Christopher Rickett, CCS-1; Dan Quinlan, Lawrence Livermore National Laboratory; Matthew Sottile, Univ. of Oregon

The Fortran Development Tools (FDT) project is dedicated to delivering productivity-enhancing and program-correctness tools to the scientific application developer. The FDT project also provides an important vehicle for source-to-source transformations, enabling research in the mapping of high-level language constructs onto advanced computer architectures like Roadrunner. The goal of this research is to allow the programmer to "write once," while relying on the source-to-source compiler to target the disparate variety of computer architectures that are available.

Because of the increasing complexity of scientific applications and the computing hardware on which they run, there is a need for tools that support the entire life cycle of application development. These tools include those that support the production and maintenance of code, tools that aid in improving the quality of code, and tools that help developers to both understand and improve application performance. This document describes work begun on a Fortran open-source compiler infrastructure created to support tool development; an infrastructure designed from the beginning for program analysis rather than back-end code generation.

In addition, we also describe a solution to a special need facing today's application developers in the DOE complex. The need is for transformational tools that support the migration of important DOE applications to run efficiently across a wide range of new computer architectures, including those with multiple homogeneous (and especially heterogeneous) processing elements. Tools are needed to reduce the costs associated with transforming software, and perhaps more importantly, to guarantee that any changes made won't affect the output of the software.

Compiler Infrastructure. The FDT project has created an open Fortran 2003 parser (see http://sourceforge.net/projects/fortran-parser/).This parser offers five advantages: 1) it is based on a language grammar (ANTLR) that allows project outsiders to relatively easily understand and modify the parser to add new language constructs, 2) it is open source, 3) it has a clean separation of the parser from middle-end consumers and Java and C interfaces through which anyone can implement custom abstract syntax trees (ASTs) and tools, 4) it is fully Fortran 2003 compliant, and 5) it has been integrated with the Lawrence Livermore National Laboratory (LLNL) ROSE, C, and C++ infrastructure (see Fig. 1).

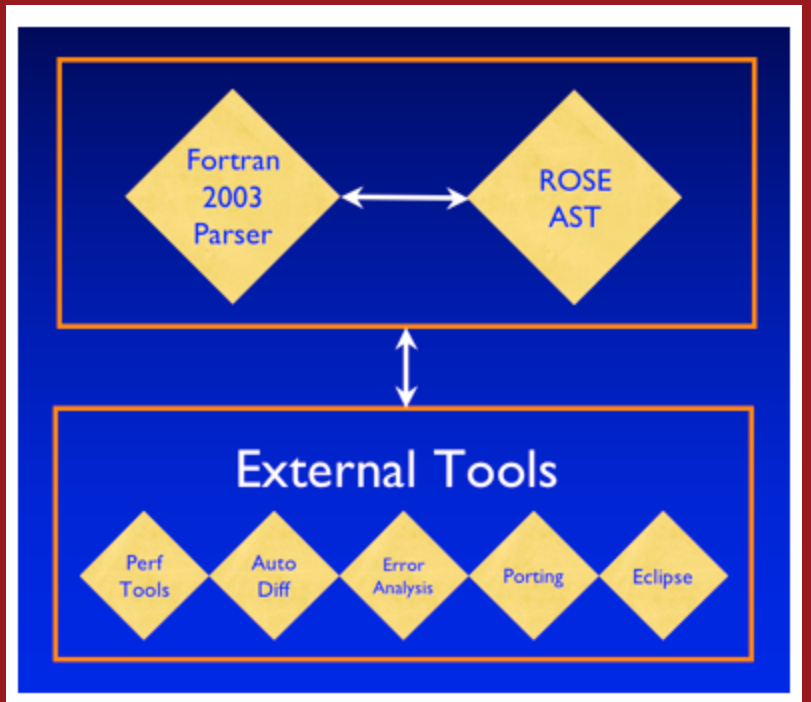

Fig. 1.The architecture of the FDT compiler infrastructure shows the clean separation of the parser, the ROSE AST (compiler middle-end), and external tool consumers.

Program Transformations. Fortran is an important high-level language because it is the language of scientific computation and also because it has many data-parallel constructs. Data parallelism is a high-level abstraction that is, at the same time, both easier to program and gives the compiler more leeway in retargeting a program to different computer architectures.

For example,

$$A = B + s * C \qquad (1)$$

is a data-parallel assignment statement where A, B, and C are arrays, and s is a scalar. Note that no explicit iteration over array indices is needed and that the individual operators, plus, times, and assignment are applied by the compiler to each individual array element independently. Thus the compiler is able to spread the computation in (1) across any hardware thread under its control.

While (1) is a very simple example, complete and very concise and elegant programs can be built up with similar statements and intrinsic functions like the array constructors (CSHIFT, MERGE, TRANSPOSE, ...), the array location functions (MAXLOC and MINLOC), and the array reduction functions (ANY, COUNT, MINVAL, SUM, PRODUCT, ...). In addition, with MPI implementations of these intrinsic functions, programs written in this data-parallel subset of Fortran are implicitly parallel; all inter-processor communication is accomplished by these library routines, not by the application programmer.

Microsoft has demonstrated (via the Accelerator project) that scientific applications written in a data-parallel language can run on traditional processors and achieve up to a 17 times speedup on graphical processing unit (GPU) processors, without recoding. The precise amount of speedup obtained depends on the scientific application and the algorithms used.

We are evaluating the use of the FDT/ROSE compiler infrastructure to transform the LANL Pagosa application to run on Roadrunner and other advanced architectures. The Pagosa code was chosen for the evaluation because it uses the data-parallel constructs of Fortran discussed above. The FDT source-to-source compiler was modified to target the IBM Cell Broadband Engine (Cell BE) processor and tested on a small portion of the Pagosa code. Input to the compiler was Pagosa Fortran, and output was C language code employing calls to the IBM ALF library and Cell BE vector intrinsics.

The results of the initial evaluation were promising. The compiler was able to automatically parallelize and vectorize the chosen section of Pagosa code. Run time was ten times faster on the Cell BE processor than on a single-core Opteron. While these initial results were promising, a larger section of Pagosa (employing the MPI CSHIFT library routines) must be tested before the evaluation is complete.

FDT Project Goals. The goals of the FDT project in the immediate future are to complete the coupling of our Fortran 2003 parser with the LLNL ROSE compiler infrastructure and then to develop tools using this infrastructure. In addition, we intend to continue our research in program transformations for advanced computer architectures. This research will provide guidance to developers on ways to program for portability across disparate computer architectures.

For more information contact Craig E. Rasmussen at crasmussen@lanl.gov.

Funding Acknowledgments

- Department of Energy, National Nuclear Security Administration, Advanced Simulation and Computing Program